

The Pennsylvania State University
APPLIED RESEARCH LABORATORY
P.O. Box 30
State College, PA 16804

User's Manual for
USURP
Unique Surfaces Using Ranked Polygons
(Version 2.32)

by

David A. Boger

Technical Report No. TR 06-005
June 30, 2006

Supported by:

NASA Grant NNJO5HA 30P

Approved for public release; distribution unlimited

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) 30-06-2006		2. REPORT TYPE		3. DATES COVERED (From - To)	
4. TITLE AND SUBTITLE User's Manual for USURP Unique Surfaces Using Ranked Polygons (Version 2.32)				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER NNJO5HA 30P	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) David A. Boger				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Applied Research Laboratory The Pennsylvania State University P.O. Box 30 State College, PA 16804				8. PERFORMING ORGANIZATION REPORT NUMBER 06-005	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) NASA/JSC Mail Code BD35 2101 NASA Parkway Houston, TX 77058				10. SPONSOR/MONITOR'S ACRONYM(S) NASA	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES 20	19a. NAME OF RESPONSIBLE PERSON David A. Boger
a. REPORT U	b. ABSTRACT	c. THIS PAGE U			19b. TELEPHONE NUMBER (include area code) 814-863-3055

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

Table of Contents

1. Background.....	1
2. Code Execution.....	2
3. Input Files	7
4. Output Files.....	8
5. Notes on Execution Modes for OVERFLOW	10
6. Post-Processing.....	11
7. Common Pathologies	12
8. Compiling USURP	14
9. References.....	16

1. Background

The purpose of USURP is to enable surface integration on grid assemblies that include overlapping surface grids. The code reads in the overset assembly, removes any overlapping portions, and can integrate the solution on the resulting surface. Users who are familiar with NASA-distributed codes such as OVERFLOW and the Chimera Grid Tools packages will recognize USURP as an alternative to FOMOCO, encapsulating the abilities of both MIXSUR and OVERINT into one executable.

USURP is motivated by the “Polymixsur” algorithm presented by Laurence Wigton at the 7th Symposium on Overset Composite Grids & Solution Technology at Huntingdon Beach, CA, in 2004, which in turn was motivated by the sometimes troublesome triangulation to form a “zipper grid” in MIXSUR. The present implementation was also motivated by the need to accommodate a broader range of CFD solvers and so was designed to accommodate both cell-centered and vertex-based codes and both structured and unstructured grids.

In the current version, seamless operation is provided for users of four CFD codes – OVERFLOW, OVER-REL and UNCLE-M (both versions of UNCLE), and CFD-SHIP. As such, no new input data formats or files are defined for use with USURP; instead, USURP will gather the information it needs from the files normally associated with the pertinent CFD code. Further details on input data and code usage are provided below. Output is also environment-dependent but normally consists of a Tecplot data file for visualization of the resulting surface assembly (see Figure 1, for example) and a list of the surface panel weights computed by USURP. The latter file, called `panel_weights.dat`, can then be read into the CFD solver for incorporation into the native surface integration procedures. Other output files associated with the different CFD environments are described in more detail below.

Because certain parts of the code are still under development, USURP is littered with an array of command line options to provide diagnostic information and to facilitate user control and experimentation. In some cases, the command line options can trigger unstable behavior or gain access to under-developed portions of the code. The default settings are intended to provide the most robust execution, thus minimizing the need for any command line options. No command line options should be needed in the majority of cases.

2. Code Execution

The full usage for `usurp` is as follows:

```
usurp [--basis=<panel,patch>] [--check-panels=<p1,p2>] [--check-patches=<p1,p2>]
      [--color] [--debug] [--disjoin=<yes,no>] [--dumptree]
      [--free-memory] [--full-surface] [--help] [--ignore-solution] [--inflate=<eps>]
      [--input=<path>] [--int=<cp,p>] [--keep-weights] [--min-factor=<eps>]
      [--min-size=<eps>] [--never-skip[=<clip,tri>]] [--notree] [--output=<path>] [--showcpu]
      [--show-panel=<panel>] [--tecformat=<ascii,binary>] [--trap[=<clip,tri>]]
      [--ttype=<GPC,Triangle,none>] [--use-priority-pairs] [--verbose[=index] (or -v)]
      [--watertight]
```

Description:

--basis=<panel,patch>

Choose basis for polygon ranking (panel or patch). When *panel* is chosen, the polygon ranking is based on the original panel area, with smaller panels outranking larger ones. When *patch* is chosen, the polygon ranking is based on the ranking of the patches to which the panels belong. The higher-ranking patch is the one with more panels overlapping the other patch; however, some patch relationships are ignored in order to create an acyclic system. The default value depends on the code environment. For OVERFLOW users, the default is *patch*; for other users, the default is *panel*. (See Figure 2)

--check-panels=<p1,p2>

Explain how two panels interact, i.e. whether or not they are considered to overlap and why. This is intended for debugging purposes and causes the code to stop once the explanation has been given.

--check-patches=<p1,p2>

Explain how two patches interact when `--basis=patch`. This is intended for debugging purposes and reports the initial relative rankings of the two patches and any changes incurred during the linear extension of the graph.

--color

Apply a 6-color vertex coloring heuristic to the patch overlap graph for visualization of the patches in Tecplot. If `--tecformat=ascii`, the zone colors are assigned in the zone headers; if `--tecformat=binary`, a separate Tecplot macro (`color.mcr`) is written, and the macro must be played in Tecplot in order to apply the revised coloring.

--debug

Display *extremely verbose* debugging information for the R-Tree sort and search, which is definitely *not* recommended.

--disjoin=<yes,no>

Specify whether different components interact. The default value depends on the code environment. For OVER-REL users, the default is *no*, which means that differently named surface components in bc.in will interact. For OVERFLOW users, the default is also *no*, which is *inconsistent* with the default behavior of MIXSUR. CFD-SHIP users currently have no mechanism for separating components but may want to consider converting their data into FOMOCO input formats in order to take advantage of this and other features.

--dumptree

Generate a (possibly very large) Tecplot file of the R-Tree, which is definitely *not* recommended.

--free-memory

Deallocate memory whenever possible as the code proceeds. This can be time consuming and does not significantly alleviate the memory demands of the code, so it is mainly intended for debugging purposes. In that case, the code will report an error if there is a net number of allocate/deallocate statements during execution, which can help in the prevention of memory leaks.

--full-surface

Triangulate the entire surface without attempting to resolve hanging nodes. (Compare this to the --watertight option.)

--help

Display a brief summary of these command line options and descriptions

--ignore-solution

Ignore any existing solution files and simply process the grid

--inflate=<eps>

Multiply all three dimensions of the axis-aligned panel bounding boxes by *eps* after --min-factor and --min-size have been applied.

--input=<path>

Specify a pathname for the main working directory containing the input files. Default value is the current directory. This option is mainly useful for processing files owned by other users.

--int=<cp,p>

For OVERFLOW users; specify whether or not to subtract the freestream pressure during the pressure integration. (The freestream pressure is

subtracted by default or if `--int=cp`, unless the solution is specified as incompressible.)

--keep-weights

Read the panel weights from `panel_weights.dat` and do *not* re-compute them. For OVERFLOW and OVER-REL users, the purpose is to re-compute the forces and moments using previously computed panel weights, which might be desirable if the geometry has been rotated, for example. (Note that the computations performed by USURP depend on the orientation of the geometry relative to the Cartesian axes.)

--min-factor=<eps>

Specify the minimum Cartesian dimension allowed for panel bounding boxes to be the product of min-factor and the diagonal length across the initial panel bounding box. Default value is 0.01 (or 1% of the bounding box size). USURP ultimately uses the larger of this dimension and the one resulting from the `--min-size` option.

--min-size=<eps>

Specify the minimum Cartesian dimension allowed for panel bounding boxes. Default value is 0 (earlier versions of the code had a default value of 1e-8). This option provides a simple approach for resolving overlaps on flat surfaces where the two grids are separated by small amounts. USURP ultimately uses the larger of this dimension and the one resulting from the `--min-factor` option.

--never-skip[=<clip,tri>]

Never skip triangulation or polygon clipping operations, even if USURP has warned that attempting a particular operation may cause the code to crash. Selecting `--never-skip=clip` will cause the code to repeat any failed polygon clipping operations while skipping any dangerous triangulations. Selecting `--never-skip=tri` will cause the code to attempt all triangulations but not repeat any failed polygon clipping operations. Using `--never-skip=clip,tri` or `--never-skip` alone causes the code to never skip any operation.

--notree

Disable the use of the R-Tree in finding overlapping surface panels. This option is provided mainly for debugging purposes in the case where a bug is suspected in the relatively complicated, pointer-based tree sort and search. This option is *not* recommended, in part because the alternative, simple search method can be substantially more time-consuming and in part because the R-Tree coding *appears* to be stable and robust.

--output=<path>

Specify a pathname where all output files will be written. Default value is the current directory. This option is mainly useful when running the code in a folder where the user does not have write permission.

--showcpu

Display CPU times for each step of execution, with output to STDERR.

--show-panel=<number>

Plot the evolution of a specified panel (suitable for subsequent animation using Tecplot). The default is to show no panel. Output is a series of files named *slide*.dat*, which can be animated by modifying and running the Tecplot macro *MakeMovie.mcr* that is provided along with the USURP source code.

--tecformat=<ascii,binary>

Choose *ascii* or *binary* for the Tecplot output file. The *binary* option is only available when the Tecplot I/O library (*tecio.a*) has been linked during compilation and when the compile option `-DUSE_TECPLOT` is included, both of which can be accomplished by defining the environment variable `TECIO_HOME` to point to the Tecplot library. When `USE_TECPLOT` is defined, the *binary* option is the default.

--trap[=<clip,tri>]

This option can slow the code dramatically and is used only to investigate the cause of code crashes that can occur while clipping polygons in the GPC library or triangulating polygons in the Triangle library. The code outputs a file called *scratch_clipping.dat* just before each clipping operation (if `--trap=clip` is used) and a file called *scratch_triangle.dat* just before each triangulation (if `--trap=tri` is used), and each file is deleted if the operation is successful. In that way, a record of the offending operation is provided in the event of a crash. Specifying “`--trap`” is the same as specifying “`--trap=clip,tri`”.

--ttype=<GPC,Triangle,none>

Choose *GPC* or *Triangle* for the triangulation method, or choose *none* in order to skip triangulation altogether. The *Triangle* method is preferred (and is the default option) because it does not add any new nodes during the triangulation; however, the *Triangle* library appears to be much more sensitive to degenerate input sets and can often fail, resulting in segmentation faults. The *GPC* option generates simplistic “tristrips” while adding extra nodes to the grid and is available primarily as an alternative in case the *Triangle* routine fails.

--use-priority-pairs

For OVERFLOW users; over-rule rankings with set priority pairs.

--verbose[=*index*] (or -v, -vv)

Provide extra diagnostic information to STDERR. Values for *index* may be 0, 1, or 2, with increasing values providing increasing amounts of diagnostic output. An index of 1 (or -v) typically leads to increased screen output, including cpu time spent in each routine (i.e. --showcpu is automatically invoked), information regarding the outcome of the patch-based ranking system, and an explicit list of any polygons for which triangulation was skipped. An index of 2 (or -vv) further increases the screen output (for example, the reason why a particular triangulation was skipped is given) in addition to providing some diagnostic output files. For example, the file skipped.dat is a Tecplot file containing the polygons for which triangulation was skipped.

--watertight

Create a watertight triangulated surface by triangulating the entire surface and resolving any hanging nodes introduced by the polygon clipping operation. Triangulation by “Triangle” is required.

3. Input Files

No special input files are needed for USURP.

For OVERFLOW users, USURP will read the grid file (which must be named `grid.in`) as well as input from STDIN. The input data stream should follow the same conventions as the standard input for MIXSUR. Priority pair data is ignored by USURP by default but can be taken into consideration by adding the `--use-priority-pairs` option to the command line. If the flow file `q.save` exists, USURP will read it and use it to conduct the force and moment integration. The grid file (`grid.in`) must be unformatted but can be single- or double-precision, and multiple- or single-grid PLOT3D format. The grid file can include I-blank data, but if it does not, then I-blank is assumed to be equal to 1 for all points. The flow file (`q.save`) can be formatted as a PLOT3D q-file or OVERFLOW solution file; however, it must otherwise have the same format as the grid file (`grid.in`) as far as whether the data is in single or double precision and whether the file is in multiple- or single-grid PLOT3D format.

For OVER-REL users, USURP should be able to obtain any information it needs from `over_rel.nml` and `bc.in`. It generally expects a complete OVER-REL working directory, including grid (G*) files, restart (RST*) files, eddy viscosity (MUT*) files (for best results), and distance (D*) files (if IWFN=1). For overset cases, the SUGGAR DCI file will also be needed.

For CFD-SHIP users, USURP should be able to obtain any information it needs from `cf_ship.nml`. It generally expects a complete CFD-SHIP working directory, including a grid (*grfd) file and boundary condition (*bcs) file. For overset cases, the Pegasus XINTOUT file is also needed.

For UNCLE-M users, USURP should be able to obtain any information it needs from `uncle.inp` and the `uncle.grd*` grid files. For overset cases, the SUGGAR DCI file will also be needed.

4. Output Files

- a. **usurp-surfaces.plt**: Tecplot binary file for viewing the surface panel data. (Alternatively, if `--tecformat=ascii` is specified on the command line, then `usurp-surfaces.dat` is output instead, which is useful mainly when the Tecplot I/O library is not available or for debugging purposes when the Tecplot binary file is corrupt.)
- b. **panel_weights.dat**: Data file containing the computed weights of each surface panel, for use as input data to the CFD solver, where the panel weights can be accommodated directly in the native force and moment integration routine. The data is an ASCII, formatted file, with the weights written using one record for each grid patch or surface subset, in the order that the patches are encountered on input. Within each patch, the data is written using I,J,K nested loops (I running fastest) *regardless of the orientation of the surface*.
- c. **usurp_data.txt** (OVER-REL users only): Excel file containing pressure, viscous, and total forces and moments on each component. Moments are measured about the point (xcg,ycg,zcg), which is obtained from `over_rel.nml`.
- d. **STDOUT**: Some basic code diagnostic information is written to STDOUT. In addition, for OVERFLOW users, the pressure, viscous and total forces and moments for each integration subset and each integration surface is written to STDOUT, consistent with output generated by OVERINT. As with OVERINT, the origin for the moments is specified through STDIN in the MIXSUR input format.
- e. **STDERR**: Information written to standard error consists of fatal error messages as well as CPU time information when the “`--showcpu`” command line option is used. USURP checks for dozens of run-time errors, which it may report. If such errors occur, please report them to dab143@only.arl.psu.edu.
- f. **mixsur.fmp** (OVERFLOW users only): ASCII file containing a summary of the MIXSUR-style input, including integration subsets, surfaces, and components, and reference conditions. This file is required for input to OVERFLOW when using USURP in “OVERFLOW mode” (see Section 5).
- g. **grid.i.tri** (OVERFLOW users only): ASCII file containing the triangulated surface representation, provided only when the `--full-surface` or `--watertight` options are included on the command line. If the solution file is present, the file will instead be named **grid.i.triq** and include the conservative variable data.
- h. **grid.ibi** (OVERFLOW users only): unformatted file containing the grid and i-blank information for the integration subsets. This file is required for input to OVERFLOW when using USURP in “OVERFLOW mode” (see Section 5). The file is either single- or double-precision, consistent with the precision of `grid.in`.

Presently, the i-blank values are set to one everywhere since it is assumed that the USURP panel_weights.dat file is all that is needed to carry out the surface integration correctly.

- i. **color.mcr**: Tecplot macro that applies a 6-color graph coloring heuristic to the zone mesh colors of usurp-surfaces.plt. The macro is only provided when the USURP options --color and --tecformat=binary are both specified. If --tecformat=ascii is specified, the color attributes are written directly into the zone headers in the usurp-surfaces.dat file.
- j. **usurp-triangles.plt**: Tecplot binary file for viewing the triangulated surface data, provided only when the --full-surface or --watertight options are specified on the command line. (Alternatively, if --tecformat=ascii is specified on the command line, then usurp-triangles.dat is output instead, which is useful mainly when the Tecplot I/O library is not available or for debugging purposes when the Tecplot binary file is corrupt.)

5. Notes on Execution Modes for OVERFLOW

The “stand-alone mode” for USURP is accomplished by using USURP as a *post*-processing code. In this instance, USURP will look for a grid file (grid.in) and accept the traditional MIXSUR input from the standard input. If the solution file (q.save) exists, USURP will read it and generate the forces and moments; otherwise, it will simply generate the panel weights.

The “OVERFLOW mode” for USURP is accomplished by using USURP as a *pre*-processing code to generate the panel weights file (panel_weights.dat), as well as the MIXSUR parameter file (mixsur.fmp) and an integration i-blank file (grid.ibi). OVERFLOW will recognize and read the panel weights file and apply the panel weights to its native surface force and moment integration. Note that the integration i-blank file presently sets all i-blank values to 1 – any regions of the grid that should be blanked out by the i-blank values should have panel weights of zero set automatically by USURP.

6. Post-Processing

In Tecplot, load `usurp-surfaces.plt` (the last zone contains the triangulated regions) and play the Tecplot macro “`setup.mcr`” (provided in the USURP source code directory; frequent users may wish to add this to Tecplot’s “Quick Macro” panel). If `setup.mcr` is not available, set the contour variable to **iblack** and set the contour type for all zones to “primary value flood.” Enable I-Blanking for any primary value whose **ratio** is less than 0.999. Turn on Transparency. Turn the boundaries off. At this point, you should see a quasi-water-tight solid surface with no overlapping (Figure 1, for example). If overlapping areas remain, they will show up darker due to the transparency of the surface. The surface should be uniformly colored, showing that all visible panels have an **iblack** value of 1. In rare cases, a panel having an **iblack** value less than 0 (fringe) or equal to 101 (orphans) may be partially visible. This would only occur in regions of minimal overlap.

Very small gaps will inevitably occur in triangulated regions. This is to be expected and is an issue that arises only in the visualization of the result. It is important only that the gaps be much smaller than the size of the neighboring surface panels.

Errors, if they occur, would manifest themselves as large gaps (implying too much of the surface was removed) or large overlapping regions (implying not enough of the overlapping grid regions were removed). In either case, “large” means “on the order of the neighboring surface panel areas” (say, 5% or more of the local panel areas). Tip: If the problem appears to be due to separate geometric components in very close proximity (e.g. a hull and appendage with a very small gap between them), use the command line option “--disjoin” to prevent this interaction.

OVER-REL and OVERFLOW users may wish to compare the integrated forces and moments to those from the native integration procedures. By defining “components” (OVER-REL) or “integration surfaces” (OVERFLOW) consisting entirely of non-overset grids, a direct comparison between USURP and the native procedures can be obtained. Identical results should be achievable in non-overset regions.

7. Common Pathologies

- a. Large gaps in the surface – Gaps in the surface may be due to a panel with an i-blank value of zero that is not covered by panels with i-blank values not equal to zero. The problem can be diagnosed in Tecplot by turning off the i-blanking and coloring the panels by the contour variable “i-blank.” As with all scalar variables, the contour attributes should be set to “primary value flood.”
- b. Overlapping region remains – Overlapping regions may remain when a pair of *apparently* overlapping panels fails to form a *valid* overlapping pair. Most often, this is because the bounding boxes of the two panels fail to overlap. (To find the exact cause for a particular pair of panels, use the --check-pair option in USURP.) The panel bounding boxes are aligned with the Cartesian axes, so a panel that is perpendicular to one of the Cartesian axes will have zero thickness in one dimension. Therefore, if two panels are intended to lie on the same plane but are in fact lying on parallel planes a small distance apart, their bounding boxes may not overlap. The situation can be remedied using the command line option --min-size=<eps> where “eps” specifies the minimum thickness of the panel bounding boxes. Alternatively, set --min-factor=<eps>, which sets the minimum thickness to be “eps” times the length of the bounding box diagonal. Naturally, setting the value too large can cause bounding boxes to overlap that should not.
- c. Small gaps in the surface – Small gaps in the surface (gaps much smaller than the size of the adjoining panels) are to be expected and are a natural consequence of the algorithm. Panels are projected onto a plane prior to undergoing polygon clipping and are rotated back afterwards. The small gaps will be more visible when the four vertices of the adjoining panel are increasingly far from being coplanar.
- d. Triangles overlapping full panels – Unexpected results are often a result of an interaction between *apparently* non-overlapping surfaces. Panels will interact when their bounding boxes overlap. Because the bounding boxes are axis-aligned, panels oriented at angles relative to the Cartesian axes may have large bounding boxes. Panels whose bounding boxes thus overlap may clip one another if the dot product of their surface normals is greater than *zero*. Therefore, a tapered control fin may interact with the body from which it extends since the taper causes the dot products of the surface normals to be slightly greater than zero. One way to identify the cause of such problems in Tecplot is to color the panels by the contour variable “ipan,” which can help to identify the source of the intruding panel. Alternatively, the Tecplot probe tool can be used to identify the panel numbers and the “--check-pair” option in USURP can then be used to discover the cause of the problem. The problem can be corrected by separating the surfaces into different components and using the command line option “--disjoin” or “--disjoin=yes”.

e. Segmentation faults:

If using Intel Fortran compilers, be sure to set “ulimit –s unlimited.” Intel compilers starting with version 8 use more stack for storage of temporary arrays, requiring care that enough stack is available.

If the segmentation fault occurs during the GPC clipping (after the R-tree is built and while the code is within the GPC routines), then try adjusting GPC_EPSILON (in gpc.h), either by raising it or lowering it by an order of magnitude or two. Increasing the value encourages merging of points, which helps prevent lots of tiny elements from being left over in the final surface. The library was written with the tolerance very small, however, and increasing it might be *causing* the robustness problem. Setting --trap=clip can isolate the polygon clipping operation that is causing the failure by writing a file called scratch_clipping.dat prior to the Boolean difference operation for each pair, but note that this option can dramatically increase the time it takes to run the code.

If the segmentation fault occurs during the triangulation when Triangle is used, then there is a good chance that the polygon data being passed to Triangle is degenerate or does not otherwise meet Triangle’s demanding standards. Setting --trap=tri can isolate the problematic polygon by writing a file called scratch_triangle.dat prior to triangulating each polygon, but note that this option can dramatically increase the time it takes to run the code. The GPC tristrrips seem to be more lenient in this respect, so switch to --ttype=GPC if possible or else modify the “GarbageCollector” routine to clean up the polygon. Specifying --ttype=none will avoid the triangulation altogether.

If segmentation faults are proving more of a problem than run time, then it is probably worthwhile to always compile the code with the debugging flags so that when a problem does arise it can be tracked down. Often, a problem that arises while the code is optimized may disappear again when the code is recompiled with the debugging options on.

8. Compiling USURP

- a. To generate the executable “usurp” type “make <machine> [CMD=<cmd>]”. A list of valid machine names and commands can be viewed by typing “make” or “make help”. Note that the machine-dependent compiler options are entered in Make.sys, which is included into Makefile. Also note that Makefile is generated by the Perl script makemake.pl. To regenerate the Makefile (in order to update dependencies), type “makemake.pl usurp”. Note that the code is always compiled in double precision, even for single-precision input/output.
- b. USURP is written using (mostly) standard Fortran 90 and is linked to the General Polygon Clipping (GPC) and Triangle libraries, which are written in C. If the environment variable TECIO_HOME is defined, USURP also makes calls to the Tecplot I/O library (tecio.a), allowing the code to write a binary Tecplot file. The code should be compiled with “-DMCLOCK” when the intrinsic function “mclock” is available, or with “-DCPU_TIME” when the intrinsic subroutine “cpu_time” is available. Byte swapping options are normally specified through the compiler options (such as -Mbyteswapio under pgf90) but can be specified through the use of -DCONVERT_BIG_ENDIAN when the “convert” attribute of “open” is available.
- c. Binary Tecplot output files can be generated by linking USURP with the tecio.a library provided with Tecplot. Since the location of the Tecplot library varies from one system to another, the link can be accomplished by defining the environment variable TECIO_HOME, e.g. “setenv TECIO_HOME /tecplot/path/tecio.a”.

The non-standard %VAL intrinsic that is used in the code is associated with the binary Tecplot output. If the command is a problem, remove Tecplot from the compilation (i.e. eliminate TECIO_HOME; see preceding paragraph).

If compilation results in the error “undefined reference to __gxx_personality_v0”, add -lstc++ and/or -lsupc++ to the TECIO_HOME definition or remove Tecplot from the compilation (see two preceding paragraphs). To add the c-libraries on the command line, use “make <machine> LIBS=-lstc++ -lsupc++”. If the TECIO_HOME variable has been set (see above), then it may be better to change the TECIO_HOME definition to include the c++ libraries, e.g. setenv TECIO_HOME “/home/local/tecplot/lib/tecio.a -lstc++ -lsupc++”.

Ben Kirk, at NASA-JSC, has pointed out that the “undefined reference to __gxx_personality_v0” are errors related to a C library called “ctypeb”, which should have been statically linked to tecio.a but wasn’t. Here’s how to modify tecio.a to fix these unresolved references:

```
cp $TECIO_HOME .
gcc -c ctype.c
```

```
ar -qv tecio.a ctype.o
make intel8 "LIBS=./tecio.a"
```

A copy of ctype.c has been provided in the EXTRAS subfolder of the USURP source.

- d. USURP relies on the General Polygon Clipping Library (Version 2.32) by Alan Murta for polygon Boolean operations and triangulation. The GPC library is free for non-commercial use. Anyone wishing to use the GPC library in support of a commercial product should email gpc@cs.man.ac.uk. (A full copyright notice is included in gpc.c in the GPC2.32 subfolder below the USURP source code.)

The GPC library uses a tolerance for merging points, which is set in gpc.h. The original code uses "DBL_EPSILON" for this tolerance, which can cause very small triangles to be leftover at the end of USURP. Raising the tolerance to 1E-06 eliminates this but can lead to segmentation faults (particularly ahead of "merge_right" and "merge_left") in the gpc_polygon_clip module in gpc.c. Some error handling has been added to skip these polygons in these cases and issue a warning message, but if such warning messages persist, the tolerance in gpc.h should be addressed. To avoid crashes, Murta recommends keeping the tolerance as small as possible.

- e. USURP also relies on Triangle (Version 1.6) by Jonathan Shewchuk for triangulation. Please note that although Triangle is freely available, it is copyrighted by the Jonathan Shewchuk and may not be sold or included in commercial products without a license. (A full copyright notice is included in the README file of the TRIANGLE1.6 subfolder below the USURP source code.) Specific instructions for compiling and using the Triangle library are in the TRIANGLE1.6 subdirectory.

The Triangle library, which is written in C, has extensive documentation which has been copied to the subfolder TRIANGLE1.6. Included in the documentation is advice on compiling the library, which recommends defining REDUCED and CDT_ONLY to reduce the size of the library object file, SELF_CHECK to have the routine report diagnostic error messages, and LINUX for proper behavior on Linux machines. These definitions are included in CFLAGS in Make.sys.

- f. Compiler options for the Portland Group compiler (pgf90) may include -Mbyteswapio -Mstandard -O2 -DUSE_TECPLOT and -DMCLOCK (as described above).
- g. Note that pgcc (Portland Group) is unable to compile the Triangle library when the LINUX flag is defined due to a syntax issue in /usr/include/fpu_control.h in the macro FPU_SETCW. Compilation should be successful using gcc, however.

9. References

Boger, David A., and Dreyer, James J., “Prediction of Hydrodynamic Forces and Moments for Underwater Vehicles Using Overset Grids,” AIAA Paper 2006-1148, 44th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January 2006.

Chan, William M., and Buning, Pieter G., “User’s Manual for FOMOCO Utilities – Force and Moment Computation Tools for Overset Grids,” NASA Technical Memorandum 110408, July 1996.

Guttman, Antonin, “R-Trees: A Dynamic Index Structure for Spatial Searching,” Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 47-57, 1984.

Murta, Alan, “A General Polygon Clipping Library,” Version 2.32, Advanced Interfaces Group, Department of Computer Science, University of Manchester, <http://www.cs.man.ac.uk/aig/staff/alan/software/gpc.html>.

Shewchuk, Jonathan Richard, “Triangle,” Version 1.6, Computer Science Division, University of California at Berkeley, <http://www.cs.cmu.edu/~quake/triangle.html>.

Wigton, Laurence B., “Polymixsur – Boeing’s Replacement for Mixsur,” 7th Symposium on Overset Composite Grids & Solution Technology, Huntington Beach, CA, October 5-7, 2004.

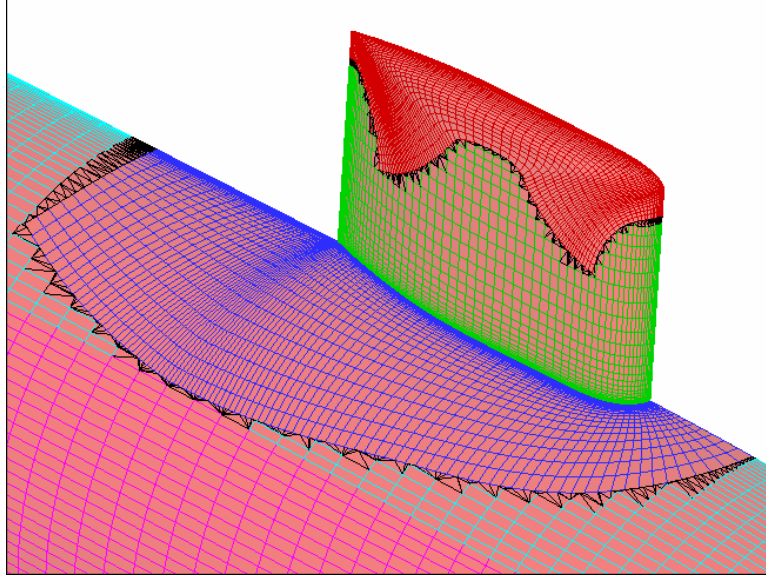


Figure 1 Surface mesh resulting from USURP. The surface is colored by **iblack** to show that only field points contribute to the integration domain.

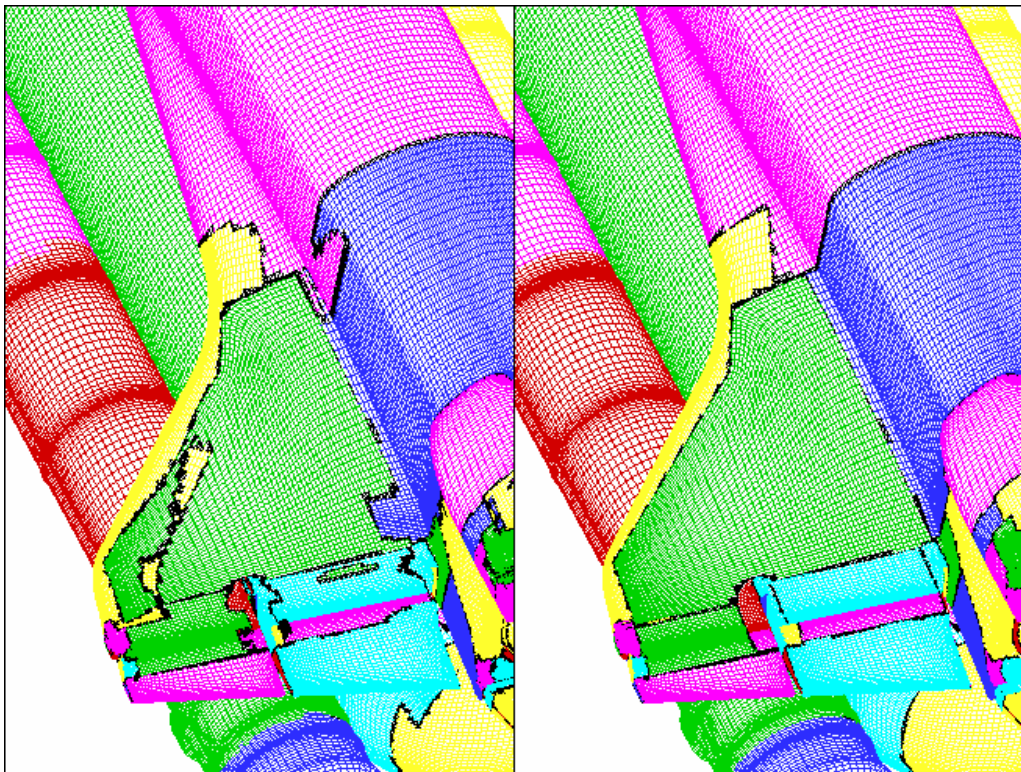


Figure 2 Comparison of results using the `--basis=panel` and `--basis=patch` options.